RF-LOGIC: SCALABLE RECENCY-FREQUENCY LOGIC RULES LEARNING FOR INTERPRETABLE TEMPORAL KNOWLEDGE GRAPH FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Temporal knowledge graph forecasting has important applications in recommendation systems, geopolitical forecasting, healthcare, and beyond. Recent works address this task by combining temporal logic rule learning with neural networks. However, existing methods often compromise interpretability when incorporating neural components and do not scale well to large temporal knowledge graphs. In this work, we introduce RF-LOGIC, a scalable and interpretable model for temporal knowledge graph forecasting that provides explanations through temporal logic rules. Moreover, we introduce two new types of temporal logic rules for temporal knowledge graph forecasting that RF-LOGIC can express: (i) symmetric shared-object rules and (ii) negation rules. In addition, we propose a dedicated training scheme for RF-LOGIC that improves generalization while preserving interpretability. To demonstrate the effectiveness of RF-LOGIC, we evaluate it on TGB 2.0 and show that it performs on par with, or exceeds the state-of-the-art methods across multiple temporal knowledge graph forecasting metrics.

1 INTRODUCTION

A knowledge graph (KG) is a mathematical structure for representing facts as triples consisting of two entities and a relation type, e.g., (Paris, is capital of, France). Since multiple relation types may hold between the same pair of entities, KGs can also be viewed as edge-labeled multigraphs. KGs give rise to several important tasks, including querying (Ren et al., 2020; Ren & Leskovec, 2020; Galkin et al., 2022) and reasoning (Luo et al., 2024; Liu et al., 2024; Cui et al., 2024). The development of strong techniques for these tasks, together with the ability of KGs to capture relational factual knowledge, has made KGs a core component of practical applications such as retrieval-augmented generation (RAG) (Sen et al., 2023; Edge et al., 2024) and search engines (Xiong et al., 2017; Liu et al., 2018).

Temporal Knowledge Graphs (TKGs) extend KGs by associating each fact with a timestamp that indicates *when* it was valid or occurred. As such, TKGs capture the evolution of relational events over time, e.g., (Napoleon, born in, Corsica, 1769). Among the central problems in this setting — TKG forecasting (Han et al., 2020; Li et al., 2021), has emerged as a particularly important and challenging task: given a history of observed temporal facts, the goal is to predict facts at future timestamps.

Common techniques for TKG forecasting include scoring potential relations based on recency (Poursafaei et al., 2022; Gastinger et al., 2024b), deriving temporal logic rules to predict future relations (Liu et al., 2022; Li et al., 2023), and embedding-based approaches that use neural networks to score potential future relations (Han et al., 2021; Li et al., 2022a). Rule-based techniques are often preferred not only for their interpretability, but also for their ability to transfer temporal logical rules to new datasets that share the same relation types. The success of temporal logic rules in predicting future relations has motivated the development of hybrid techniques that combine temporal logic rules with embedding methods (Mei et al., 2022; Liu et al., 2023; Lin et al., 2023). However, their interpretability is often compromised by the embedding components. Furthermore, rule-based, embedding-based, and hybrid approaches often fail to meet runtime and memory constraints (Gastinger et al., 2024a), limiting their practical applicability and deployment.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this work, we propose a new hybrid approach, called RF-LOGIC, that is both interpretable and efficient, satisfying the crucial runtime and memory constraints in common benchmarks (Gastinger et al., 2024a). By combining an efficiency-oriented implementation in terms of memory and runtime with novel types of temporal logic rules, our method achieves strong performance on the task of TKG forecasting while remaining practical in terms of both time and memory efficiency.

Main Contributions. Our contributions are as follows:

1. We introduce RF-LOGIC, a hybrid framework for TKG forecasting that combines a frequency-recency feature extractor for candidate future relations with temporal logical rule learning. Designed with an efficient runtime and memory oriented implementation, RF-LOGIC satisfies practical runtime and memory constraints and scales to large temporal knowledge graphs.
2. We provide new logic rules for the proposed recency-frequency feature extractor and show that RF-LOGIC preserves the interpretability advantages of rule-based forecasting while benefiting from the effectiveness of hybrid modeling.
3. We perform extensive experiments on TGB 2.0, a standardized benchmark for TKG forecasting. The results show that RF-LOGIC achieves state-of-the-art or competitive predictive performance across datasets, while meeting practical runtime and memory constraints.

2 BACKGROUND

Let $\mathcal{E}$ and $\mathcal{R}$ denote the sets of entities and relations, respectively, and let $\mathcal{T} \subset \mathbb{R}^+$ denote the set of timestamps. A temporal knowledge graph (TKG) is a set $\mathcal{G} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E} \times \mathcal{T}$. Each quadruple $(s, r, o, t) \in \mathcal{G}$ represents a fact observed at time $t \in \mathcal{T}$, where $s \in \mathcal{E}$ is the subject entity, $o \in \mathcal{E}$ is the object entity, and $r \in \mathcal{R}$ is the relation between them. Such quadruples are also referred to as edges or links in the TKG. Equivalently, we say that s is in relation r with o at time t. If the entities in the TKG are associated with a set of types $\mathcal{K}$ through a mapping function $\mathcal{F} : \mathcal{E} \to \mathcal{K}$, which assigns each entity to its type (e.g., *Napoleon* to *Person* and *Horse* to *Animal*), then the graph is called a temporal heterogeneous graph (THG). For convenience, we refer to both TKGs and THGs as TKGs, and distinguish between them only when explicitly discussing different types of datasets.

TKG forecasting is the task of predicting missing links at a future timestamp t^+, where $t^+ > t$ for all $(s, r, o, t) \in \mathcal{G}$. In this work, we focus on the specific TKG forecasting setting of *entity forecasting*, namely predicting the object entity for queries of the form $(s, r, ?, t^+)$. Entity forecasting is typically formulated as a ranking task, in which the model assigns a relevance score to each candidate entity and ranks all entities according to their likelihood of completing the query. Entity forecasting is commonly evaluated using ranking-based metrics, primarily Mean Reciprocal Rank (MRR), which averages the reciprocal rank of the ground-truth object entity, and Hits@k, which measures the proportion of queries for which the ground-truth object entity is ranked among the top-k predictions.

Temporal (cyclic) logic rule of length n consists of entity variables e_i and timestamp variables t_i for $1 \leq i \leq n+1$, where $e_i \in \mathcal{E}$ and $t_i \in \mathcal{T}$. The timestamps satisfy the temporal order constraints $t_{i-1} \leq t_i \leq t_{i+1}$ for all valid $i < n$, and in particular $t_n < t_{n+1}$. The rule further contains fixed relations $r_1, r_2, \ldots, r_n, r_h \in \mathcal{R}$, and is defined as

$$(e_1, r_h, e_{n+1}, t_{n+1}) \leftarrow \bigwedge_{i=1}^{n} (e_i, r_i, e_{i+1}, t_i). \tag{1}$$

The left-hand side is called the *head* of the rule, and the right-hand side is called the *body*. Such a rule expresses that whenever there exists an assignment of entities and timestamps satisfying the body, the head is likely to hold at a later time. Since such dependencies are generally not deterministic in real-world TKGs, temporal rule learning methods typically associate each rule with a confidence score that quantifies how reliably the rule holds based on previously observed facts in the graph.

3 METHOD

We propose RF-LOGIC, a scalable and interpretable model for TKG forecasting. RF-LOGIC scores each candidate object using two complementary sources of historical evidence observed strictly before t^+: (i) direct recent interactions from s to o, and (ii) unique shared objects recently reached by both s and o under the same relation. These two signals are summarized into an interpretable descriptor for the candidate pair (s, o). A relation-specific linear scoring function then maps this descriptor to the final score for query relation r. The resulting model is efficient, memory-bounded, and directly interpretable in terms of temporal logic rules. In Section 3.1, we provide an overview of RF-LOGIC, in Section 3.2, we detail its memory module and analyze its memory scalability, in Section 3.3, we describe the feature extraction component, in Section 3.4, we describe relation-specific scoring, and in Section 3.6, we demonstrate how to train RF-LOGIC effectively.

3.1 OVERVIEW

For each entity, RF-LOGIC maintains a fixed-size memory of its most recent outgoing interactions. At prediction time, the memories of the query subject s and candidate object o are queried to construct two feature blocks:

$$d_{s,o} = \left[d_{s,o}^{(1)}; d_{s,o}^{(2)}\right] \in \mathbb{R}^{2|\mathcal{R}|}.$$

The first block, $d_{s,o}^{(1)} \in \mathbb{R}^{|\mathcal{R}|}$, contains direct 1-hop recency-frequency features. For each historical relation r', these features measure how much of the recent relation-r' interaction mass of s points to candidate object o. The second block, $d_{s,o}^{(2)} \in \mathbb{R}^{|\mathcal{R}|}$, contains unique shared-object frequency features. For each relation r', this feature measures the symmetric frequency of distinct recent objects that appear in the outgoing memories of both s and o under relation r'.

For each query relation $r \in \mathcal{R}$, the model learns a relation-specific parameter vector

$$w_r = \left[w_r^{(1)}; w_r^{(2)}\right] \in \mathbb{R}^{2|\mathcal{R}|},$$

where $w_r^{(1)}, w_r^{(2)} \in \mathbb{R}^{|\mathcal{R}|}$ weight the 1-hop and shared-object feature blocks, respectively. The score of candidate object o for query $(s, r, ?, t^+)$ is

$$\phi(s, r, o, t^+) = w_r^\top d_{s,o} = \sum_{r' \in \mathcal{R}} w_r^{(1)}[r'] d_{s,o}^{(1)}[r'] + \sum_{r' \in \mathcal{R}} w_r^{(2)}[r'] d_{s,o}^{(2)}[r']. \tag{2}$$

Thus, the learned weights identify which temporal rule bodies support or suppress each target relation.

3.2 MEMORY MODULE

We now describe the memory module used to keep historical TKG information, how it operates and scales.

Entity memories. For each entity $e \in \mathcal{E}$, RF-LOGIC maintains a fixed-size memory of its most recent outgoing interactions before the query time. At time t^+, this memory is denoted

$$\mathcal{M}_e(t^+) = \big((r_1, x_1), \ldots, (r_k, x_k)\big),$$

where (r_i, x_i) is the relation and object of the i-th most recent interaction in which e appeared as the subject. Thus, position $i = 1$ corresponds to the most recent outgoing interaction of e before t^+. Empty memory positions are padded and ignored by the feature extractor. The memory is implemented using two GPU-resident arrays,

$$\boldsymbol{M}_{\text{rel}} \in \mathcal{R}^{|\mathcal{E}| \times k}, \qquad \boldsymbol{M}_{\text{obj}} \in \mathcal{E}^{|\mathcal{E}| \times k},$$

which store the relation and object components of the memory, respectively. When a new event (s, r, o, t) is processed, the pair (r, o) is inserted at the front of the memory of s, and the oldest entry is discarded. Memories are updated online in chronological order. Therefore, when forecasting at time t^+, all memory entries correspond to events with timestamps strictly smaller than t^+, preventing future information leakage.

Batch operations on the GPU. During mini-batch processing, we use the t-batch mechanism (Kumar et al., 2019) to avoid intra-batch collisions caused by multiple events sharing entities within the same batch. The model first scores the events in the current collision-free mini-batch using memories from the past, and only then updates the memories with the processed events. In addition to the provided code, in Appendix B we detail how to efficiently extract and process the memory for each of the two types of descriptor blocks.

Scalability. The persistent memory requires $O(|\mathcal{E}|k)$ storage. This memory footprint is linear in the number of entities and independent of the number of observed temporal edges. Consequently, RF-LOGIC can be applied to large TKGs with long histories while keeping memory consumption bounded.

3.3 FEATURE EXTRACTION

Given a query $(s, r, ?, t^+)$ and a candidate object o, the feature extractor reads the memories of both s and o and constructs the descriptor

$$d_{s,o} = \left[d^{(1)}_{s,o}; d^{(2)}_{s,o}\right] \in \mathbb{R}^{2|\mathcal{R}|}.$$

The two feature blocks are computed from the same entity memories but capture different temporal rule bodies. The 1-hop block uses the memory of s to measure direct historical evidence from s to o. The shared-object block uses the memories of both s and o. For computing the second feature block, the candidate object o is treated as an entity with its own outgoing memory, in other words, o plays the role of a subject in the body of the shared-object rule.

Relation-specific memory view. For each entity e and relation $r' \in \mathcal{R}$, let

$$\mathcal{H}_{e,r'}(t^+) = \left(x^{(1)}_{e,r'}, \ldots, x^{(n_{e,r'})}_{e,r'}\right)$$

denote the ordered list of objects appearing in the memory of e under relation r'. The order is inherited from the memory after filtering by relation, so $x^{(1)}_{e,r'}$ is the most recent object reached by e through relation r', $x^{(2)}_{e,r'}$ is the second most recent such object, and so on. The length $n_{e,r'} \leq k$ is the number of stored outgoing interactions of e under relation r'.

1-hop Recency-Frequency features. The first feature block captures direct historical interactions from s to candidate object o. For each relation $r' \in \mathcal{R}$, we define

$$d^{(1)}_{s,o}[r'] = \begin{cases} \dfrac{\sum_{i=1}^{n_{s,r'}} \mathbf{1}\left[x^{(i)}_{s,r'} = o\right] 2^{-\lambda i}}{\sum_{i=1}^{n_{s,r'}} 2^{-\lambda i}}, & n_{s,r'} > 0, \\ 0, & n_{s,r'} = 0. \end{cases} \tag{3}$$

Here, $\lambda \geq 0$ is a recency-decay hyperparameter. The rank i is relation-local: $i = 1$ denotes the most recent interaction of s under relation r', not necessarily the most recent interaction in the full memory of s. Equation 3 measures the normalized decayed mass assigned to candidate object o among the recent relation-r' interactions of s. When $\lambda = 0$, the feature reduces to the fraction of stored relation-r' interactions of s whose object is o. As λ increases, the feature assigns more weight to more recent interactions under relation r'. When $\lambda \geq 1$ the most recent object receives the largest score, regardless of the frequencies of older objects in the memory of s. Unlike approaches that compute history statistics only for the queried relation (Gastinger et al., 2024b), RF-LOGIC computes Equation 3 for every historical relation $r' \in \mathcal{R}$ and lets the relation-specific scorer learn which past relations are predictive of the query relation r.

Unique shared-object frequency features. The second feature block captures symmetric shared-object evidence. The motivation is that two entities that recently interacted with the same objects under the same relation may be likely to interact with each other in the future. For example, two users who both posted in the same discussion thread may later contact each other directly.

The corresponding temporal logic rule is described by the form

$$(e_1, r_h, e_2, t^+) \leftarrow (e_1, r', e_3, t_1) \wedge (e_2, r', e_3, t_2), \qquad t_1, t_2 < t^+. \tag{4}$$

When this rule is evaluated for a candidate pair (s, o), we instantiate $e_1 = s$ and $e_2 = o$. Thus, although o is the candidate object in the query, it is treated as a subject in the second body atom and is queried through its own outgoing memory.

For each entity e and relation r', define the set of distinct recent objects reached by e under relation r':

$$\mathcal{U}_{e,r'}(t^+) = \left\{ x \in \mathcal{E} : x \text{ appears in } \mathcal{H}_{e,r'}(t^+) \right\}.$$

The shared-object feature for candidate pair (s, o) is

$$d^{(2)}_{s,o}[r'] = \frac{1}{2} \left(\frac{|\mathcal{U}_{s,r'}(t^+) \cap \mathcal{U}_{o,r'}(t^+)|}{|\mathcal{U}_{s,r'}(t^+)|} + \frac{|\mathcal{U}_{s,r'}(t^+) \cap \mathcal{U}_{o,r'}(t^+)|}{|\mathcal{U}_{o,r'}(t^+)|} \right), \tag{5}$$

where each ratio is defined as zero when its denominator is zero. This statistic measures the symmetric frequency of unique shared objects. The first term is the fraction of s's distinct recent relation-r' objects that are also recent relation-r' objects of o. The second term measures the converse. The symmetric normalization prevents the feature from being dominated by entities with large memories and assigns high values only when the two entities share a substantial fraction of their distinct recent objects. We intentionally do not apply temporal decay to the shared-object feature. For direct 1-hop evidence, the temporal reference is unambiguous: the relation-local recency rank of the interaction from s to o. For shared-object evidence, however, each shared object is supported by two events, (s, r', x, t_1) and (o, r', x, t_2). There is no unique canonical timestamp to decay: one could use the earlier event, the later event, the time gap between the two events, or another aggregation. We therefore use a frequency over distinct recent shared objects, which keeps the feature simple, symmetric, and interpretable, and leave the design of accurate recency-aware shared-object features to future work.

3.4 Relation-Specific Linear Scoring

For each query relation $r \in \mathcal{R}$, RF-Logic learns a parameter vector

$$w_r = \left[w_r^{(1)}; w_r^{(2)} \right] \in \mathbb{R}^{2|\mathcal{R}|}.$$

Given the descriptor $d_{s,o}$, the candidate score is computed by the linear form in Equation 2. The first $|\mathcal{R}|$ entries of w_r weight direct 1-hop rule bodies, while the remaining $|\mathcal{R}|$ entries weight shared-object rule bodies. Because the model is linear in the rule-based descriptor, each learned weight is directly interpretable. A large positive weight indicates that the corresponding rule body provides evidence for the head relation r. A negative weight indicates negative evidence: activating the corresponding body decreases the score of the candidate for relation r. We interpret such weights as *negated temporal logic rules*, in the sense that the presence of the body is predictive against the head relation rather than merely uninformative. This is useful in real TKGs, where some recurring temporal patterns may reliably exclude particular future relations. The scoring function does not use bias parameters. For a fixed query $(s, r, ?, t^+)$, forecasting is a ranking problem over candidate objects. A relation-specific bias would shift all candidate scores by the same amount and therefore would not affect the ranking.

Interpretation as temporal rules. Each coordinate of $d_{s,o}$ corresponds to the activation strength of a temporal rule body. A 1-hop feature $d^{(1)}_{s,o}[r']$ corresponds to soft evidence for the direct body

$$(s, r', o, t_1), \qquad t_1 < t^+.$$

A shared-object feature $d^{(2)}_{s,o}[r']$ corresponds to soft evidence for the symmetric shared-object body

$$(s, r', x, t_1) \wedge (o, r', x, t_2), \qquad t_1, t_2 < t^+.$$

The descriptor $d_{s,o}$ therefore represents the activation strengths of all rule bodies considered by the model. The relation-specific weights w_r determine whether each activated body supports or suppresses the prediction of relation r, where suppressing weights represent negated temporal logic rules.

3.5 Extension to Higher-Order Rules

Although RF-Logic focuses on 1-hop and shared-object rule bodies for efficiency, the same memory principle can be extended to higher-order temporal rules. For a fixed path length n, one can

define features corresponding to time-respecting paths from s to o whose relation sequence matches a specified pattern $(r_1, \ldots, r_n)$. A path is time-respecting if all of its edges occur before the query time t^+ and do not decrease in time. For a relation sequence $(r_1, \ldots, r_n)$, the corresponding feature can be defined as the number of time-respecting paths from s to o matching that sequence, normalized by the total number of such paths starting from s. Adding these features would introduce an additional interpretable block of size $|\mathcal{R}|^n$, whose coordinates correspond to length-n temporal rule bodies. The associated relation-specific weights would again indicate whether each body supports or suppresses the target relation. In this work, we focus on the 1-hop and shared-object features, which already provide strong expressive power while preserving low memory cost and simple interpretation (Gastinger et al., 2024a). We leave the exploration of such higher-order temporal-rule features to future work.

3.6 Training RF-Logic

We now discuss the different training aspects of RF-Logic.

Negative sampling. For each positive quadruple (s, r, o^+, t^+) in the training sequence, we construct a set of negative objects $\mathcal{N}^-(s)$. To make training informative, negatives are sampled first from the subject-side memory of s under the queried relation, excluding the ground-truth object o^+. If no valid negative object is available in memory, we sample uniformly at random from $\mathcal{E} \setminus \{o^+\}$. The number of negatives per positive is treated as a hyperparameter.

Pairwise ranking loss. Given a positive object o^+ and sampled negative objects $o^- \in \mathcal{N}^-(s)$, we optimize the pairwise objective

$$\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{N}^-(s)|} \sum_{o^- \in \mathcal{N}^-(s)} \sigma\big(\phi(s, r, o^-, t^+) - \phi(s, r, o^+, t^+)\big), \tag{6}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Minimizing this objective encourages the positive object to receive a higher score than sampled negative objects.

The total loss for the queried relation r is

$$\mathcal{L} = \mathcal{L}_{\text{rank}} + \alpha \|w_r\|_1 + \beta \|w_r\|_2^2, \tag{7}$$

where $\alpha, \beta \geq 0$ are regularization coefficients.

Sigmoid ranking surrogate. Equation 6 is a smooth bounded surrogate for pairwise ranking error. Let

$$\Delta = \phi(s, r, o^-, t^+) - \phi(s, r, o^+, t^+).$$

Since $\sigma(\Delta)$ is monotone increasing in Δ, minimizing Equation 6 encourages the desired ranking constraint

$$\phi(s, r, o^+, t^+) > \phi(s, r, o^-, t^+).$$

We compare this objective with common logistic and margin-based alternatives in Appendix E.2.

Role of regularization. We use ElasticNet (Zou & Hastie, 2005) to regularize the ranking loss. The regularization term $\|w_r\|_1$ promotes sparsity in w_r, suppressing rule bodies that are uninformative for the target relation. This improves interpretability by selecting a small subset of active temporal rules. The $\|w_r\|_2^2$ regularization term discourages overly large weights, reducing overfitting and improving generalization.

Initialization. For every query relation r, we initialize the 1-hop weights $w_r^{(1)}$ to 1 and the shared-object weights $w_r^{(2)}$ to 0. This initialization encodes a weak prior that direct historical interactions are generally informative at the beginning of training, while shared-object dependencies should be learned only when supported by data (Poursafaei et al., 2022). In practice, this leads to stable optimization without constraining the final learned solution.

4 Experiments

In this section we evaluate the effectiveness of RF-Logic on various aspects including predictive performance, scalability, and interpretability. Following guidelines to avoid inconsistent evaluation protocols for TKG forecasting provided in (Gastinger et al., 2023), we evaluate RF-Logic on

Table 1: MRR on TGB 2.0 TKG datasets (higher is better). Boldface marks the best method.

Method	Smallpedia		Polecat		ICEWS		Wikidata	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.457	0.353	0.058	0.056	0.020	0.020	0.633	0.535
EdgeBank$_{\infty}$	0.401	0.333	0.048	0.045	0.008	0.009	0.632	0.535
RecB$_{\text{train}}$	0.639	0.605	0.203	0.198	0.270	0.211	OOT	OOT
RecB$_{\text{default}}$	0.542	0.486	0.170	0.167	0.264	0.206	OOT	OOT
CognTKE	0.523	0.497	0.285	0.277	OOM	OOM	OOM	OOM
TransFIR	0.317±0.060	0.215±0.043	0.225±0.005	0.198±0.002	OOM	OOM	OOM	OOM
RE-GCN	0.631±0.001	0.594±0.001	0.191±0.003	0.175±0.002	0.232±0.003	0.182±0.003	OOM	OOM
CEN	0.646±0.001	0.612±0.001	0.204±0.002	0.184±0.002	0.244±0.002	0.187±0.003	OOM	OOM
TLogic	0.631±0.000	0.595±0.001	0.236±0.001	0.228±0.001	**0.287**±0.000	0.186±0.000	OOT	OOT
RF-LOGIC (ours)	**0.658**±0.001	**0.635**±0.001	**0.434**±0.001	**0.424**±0.001	0.277±0.001	**0.262**±0.001	**0.646**±0.001	**0.550**±0.001

TGB 2.0 (Gastinger et al., 2024a), a standardized benchmark for TKG forecasting. We evaluate RF-LOGIC across multiple TKG benchmarks and compare it with recent state-of-the-art baselines. In Section 4.1 we detail the experimental setup, Section 4.2 presents results on TKG datasets, and Section 4.3 presents results on THG datasets, and Section 4.4 presents interpretability findings regarding RF-LOGIC. We perform an ablation study of RF-LOGIC and report it in Appendix E. Our experiments seek to address the following research questions: **(RQ1)** How does RF-LOGIC perform compared to prior art for TKG forecasting?; **(RQ2)** Does our method generalize across various types of knowledge and heterogeneous graphs?; **(RQ3)** can RF-LOGIC scale for large graphs both in terms of memory and runtime?; **(RQ4)** interpretability - can we understand how does RF-LOGIC make its decisions and what it means in terms of the examined TKG?

4.1 EXPERIMENTAL SETUP

We compare RF-LOGIC to the following baselines for TKG forecasting: EdgeBank (Poursafaei et al., 2022), TLogic (Liu et al., 2022), RecB (Gastinger et al., 2024b), TGN (Rossi et al., 2020), CEN (Li et al., 2022b), RE-GCN (Li et al., 2021), as well as the recent CognTKE (Chen et al., 2025) and TransFIR (Zhao et al., 2026). Following the evaluation protocol of TGB 2.0, we use a 70%-15%-15% chronological split into training, validation and test, over three runs, and report the MRR, Hits@1, Hits@3 and Hits@10 on all the benchmarks. Negative edges for metric computations were exactly as defined by TGB 2.0 and are shared across all baselines. According to TGB 2.0 protocol, the methods that use more than 40 GB GPU memory or run for more than a week are considered as Out Of Memory (OOM) or Out Of Time (OOT), respectively. TGB 2.0 leaderboard provides results in MRR, to include results in terms of Hits@1, Hits@3 and Hits@10 we ran all the baselines using their original code and report it in the results tables below. Results are marked with '-' if they OOM or OOT while having results from TGB 2.0 leaderboard. Models are marked with † if the results in terms of Hits@k do not match the MRR achieved on the TGB 2.0 leaderboard.

4.2 TKG FORECASTING ON TGB

To answer **(RQ1)**, we use the TGB datasets for TKG forecasting (Huang et al., 2023): `Smallpedia`, `Polecat`, `ICEWS`, and `Wikidata`. We report the results in MRR in Table 1, and in Hits@1 in Table 2. We report the results in Hits@3 and Hits@10 in Appendix C. As shown in Tables 1 and 2 **RF-LOGIC outperforms all baselines.** It improves over the best-performing model, CognTKE, by +52.2% on `Polecat`. Notably, all the baselines except for the simple EdgeBank heuristic, did not scale to Wikidata. In contrast, RF-LOGIC linear in scale memory design and fast parallel manner of score computation allows RF-LOGIC to scale to the largest TKG benchmark of TGB 2.0.

4.3 THG FORECASTING ON TGB

To answer **(RQ2)**, and **(RQ3)** we use the TGB2.0 THG datasets: `Software`, `Forum`, `Github`, and `Myket`. These datasets are larger in scale both in number of entities and quadruplets thus posing a major scalability challenge for models for TKG forecasting. Furthermore they utilize different time granularities and different number of timesteps as well as different graph characteristics (see Appendix A for more information). We report the results in MRR in Table 3, and in Hits@1 in Table 4. We report the results in Hits@3 and Hits@10 in Appendix C. Most of the baselines from

Table 2: HITS@1 on TGB 2.0 TKG datasets (higher is better). Boldface marks the best method.

Method	Smallpedia		Polecat		ICEWS		Wikidata	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.294	0.201	0.018	0.017	0.001	0.001	0.565	0.472
EdgeBank$_{\infty}$	0.233	0.169	0.013	0.013	0.001	0.001	0.564	0.472
RecB$_{\text{train}}$	0.579	0.529	0.143	0.137	-	-	OOT	OOT
RecB$_{\text{default}}$	0.466	0.400	0.120	0.116	**0.192**	0.143	OOT	OOT
CognTKE	0.484	0.440	0.202	0.200	OOM	OOM	OOM	OOM
TransFIR	0.244±0.057	0.153±0.040	0.122±0.010	0.105±0.002	OOM	OOM	OOM	OOM
RE-GCN	0.580±0.003	0.535±0.002	0.120±0.002	0.106±0.001	0.147±0.002	0.108±0.002	OOM	OOM
CEN	0.595±0.001	0.549±0.001	0.125±0.001	0.110±0.001	0.160±0.001	0.122±0.001	OOM	OOM
TLogic	0.530±0.001	0.471±0.001	0.159±0.001	0.150±0.001	0.189±0.001	0.128±0.001	OOT	OOT
RF-LOGIC (ours)	**0.620**±0.001	**0.585**±0.001	**0.331**±0.001	**0.319**±0.001	0.177±0.001	**0.163**±0.001	**0.631**±0.001	**0.535**±0.001

Table 3: MRR on TGB 2.0 THG datasets (higher is better). Boldface marks the best method.

Method	Software		forum		Github		Myket	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.279	0.288	0.534	0.534	0.355	0.374	0.248	0.245
EdgeBank$_{\infty}$	0.399	0.449	0.612	0.617	0.403	0.413	0.430	0.456
RecB$_{\text{default}}$	0.106	0.099	0.552	0.561	OOT	OOT	OOT	OOT
TGN	0.299±0.012	0.324±0.017	0.598±0.086	0.649±0.097	OOM	OOM	OOM	OOM
TGN$_{\text{edge-type}}$	0.376±0.010	0.424±0.013	**0.767**±0.005	0.729±0.009	OOM	OOM	OOM	OOM
RF-LOGIC (ours)	**0.435**±0.001	**0.498**±0.001	0.736±0.001	**0.739**±0.001	**0.449**±0.001	**0.478**±0.001	**0.431**±0.001	**0.458**±0.001

Section 4.2 could not scale to THG datasets and hence they are not reported here. As shown in Tables 3 and 4 **RF-LOGIC outperforms all baselines.** It improves over the best-performing model, EdgeBank$_{\infty}$, by +10.9% on `Software`. Notably, all the baselines except for the simple Edgebank heuristic, did not scale to both Github and Myket. Again, RF-LOGIC linear in scale memory design and fast parallel manner of score computation allows RF-LOGIC to scale to large THG datasets of TGB 2.0.

4.4 INTERPRETABILITY

To answer **(RQ4)**, we visualize the learned per-relation weights of RF-LOGIC for the TKG datasets in Figure 1 and the THG datasets in Figure 2. These matrices provide a direct view of the logic rules learned by the model. Overall, RF-LOGIC learns dataset-specific dependency patterns rather than a single universal recency heuristic, although several recurring structures emerge across datasets. In particular, datasets such as `SmallPedia`, `ICEWS`, `Wikidata`, and `Software` exhibit prominent diagonal or near-diagonal structure, indicating that RF-LOGIC often assigns high weight to the basic recurrence rules

$$(e_1, r, e_2, t^+) \leftarrow (e_1, r, e_2, t), \tag{8}$$

for every $r \in \mathcal{R}$. This observation is consistent with the solid performance of RecB on these datasets, since RecB relies only on same-relation recency. At the same time, some off-diagonal entries remain close to zero, suggesting that RF-LOGIC identifies numerous body relations as weakly informative for a given head relation, which makes its predictions easier to interpret.

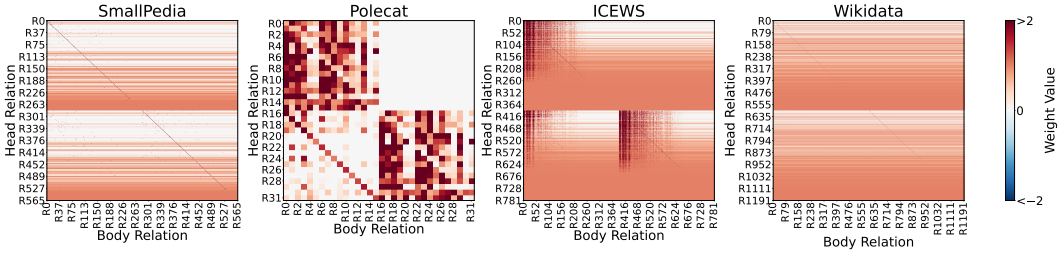

Figure 1: RF-LOGIC learned relation weights of TKG datasets.

Table 4: HITS@1 on TGB 2.0 THG datasets (higher is better). Boldface marks the best method.

Method	Software		forum		Github		Myket	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.273	0.283	0.518	0.516	0.288	0.308	0.173	0.168
EdgeBank$_{\infty}$	0.393	0.443	0.573	0.576	0.329	0.342	0.371	0.400
RecB$_{\text{default}}$	0.066	0.072	-	-	OOT	OOT	OOT	OOT
TGN†	0.016±0.003	0.019±0.002	0.300±0.020	0.229±0.034	OOM	OOM	OOM	OOM
TGN$^{\dagger}_{\text{edge-type}}$	0.091±0.004	0.112±0.005	0.300±0.020	0.229±0.034	OOM	OOM	OOM	OOM
RF-LOGIC (ours)	**0.433**±0.001	**0.496**±0.001	**0.695**±0.001	**0.697**±0.001	**0.394**±0.001	**0.426**±0.001	**0.372**±0.001	**0.402**±0.001

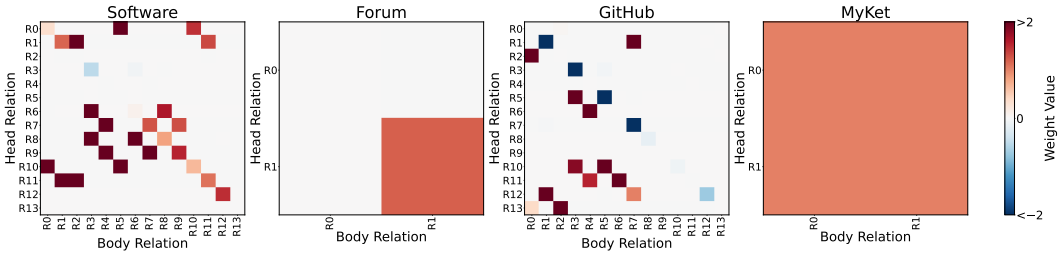

Figure 2: RF-LOGIC learned relation weights of THG datasets.

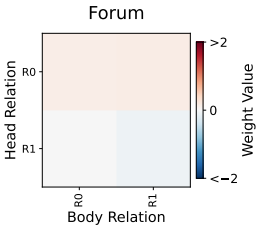

Figure 3: *Shared-object weights of Forum.*

`Polecat` exhibits a markedly different structure. Instead of a dominant diagonal, the strongest weights appear in off-diagonal blocks around the diagonal, and the relations are partitioned into two groups with strong within-group correlations and substantially weaker cross-group interactions. This pattern is readily exposed by RF-LOGIC, but it is difficult for models based only on same-relation recency, such as RecB, or on a single recency signal shared across all relations, such as EdgeBank. We further relate this structure to the underlying graph dynamics in Appendix D. `GitHub` provides another informative case. Here, RF-LOGIC learns a sparse weight matrix, indicating that only a small subset of possible rules is useful for forecasting. Notably, several diagonal entries are consistent with the negation-rule introduced in our model. In Appendix D, we show that this behavior aligns with the large fraction of "surprise" quadruples in this dataset, where subject–object pairs do not recur under the same relation. `Forum` highlights both the interpretability of RF-LOGIC and the necessity of the shared-object rule. Three out of the four learned 1-hop relation weights for `Forum` (which has two relations) are zero. The only non-zero weight is for head relation R1 with body relation R1. This means that, using only 1-hop relation weights, RF-LOGIC cannot make predictions for queries with target relation R0. The shared-object weights for R0 in Figure 3 fill this gap and guide RF-LOGIC to answer queries for R0 correctly. These weights typically correlate with frequent shared objects that appear under both relation R0 and relation R1. In Appendix D, we analyze the `Forum` dataset in greater depth and demonstrate how this learned decision-making process of RF-LOGIC correlates with the dynamic behavior of `Forum` and its domain, thereby revealing this behavior visually.

5 DISCUSSION AND SUMMARY

In this work, we introduced RF-LOGIC, a recency-frequency-based model for TKG forecasting. RF-LOGIC first computes a relevance score for the body of a logical rule and then learns the appropriate weight of the corresponding implication for the rule head. In addition, RF-LOGIC incorporates two new types of logical rules: the symmetric shared-object rule and the negation rule. RF-LOGIC is interpretable: as shown above, even a brief inspection of its weight matrices provides insight into the basis of its predictions. Moreover, we showed that the learned weights of RF-LOGIC align with the dynamic behavior of temporal graphs. Across extensive experiments, RF-LOGIC outperforms state-of-the-art baselines on multiple benchmarks and scales successfully to large graphs where other baselines fail. Notably, the improvement of RF-LOGIC over the strongest baseline reaches 10.9% on THG datasets and 52.2% on TKG datasets.

AI USE STATEMENT

In this work, we used generative AI tools for / to:

- Implement methods

We have not used generative AI tools for / to:

- Assistance in the writing of proofs.
- Help develop theoretical models or conceptual frameworks.
- Formulate mathematical claims.
- Provide critical ingredients for proving mathematical claims.
- Propose or refine hypotheses.
- Design or provide feedback on research methodology or experiments.
- Assist with translation.
- Clean and reformat dataset.
- Interpret results.
- Generation of synthetic data sets.
- Support qualitative and thematic data analysis.

Additionally, we used generative AI tools for / to:

- Create or edit software code.
- Summarize or analyse existing literature, and identify relevant literature.
- Draft parts of a research paper.
- Edit a research paper to improve readability.

We have reviewed all AI-assisted work. LLM-generated code was verified and tested rigorously for validity and correctness. All written LLM-generated text appearing in the submitted manuscript was reviewed and approved by the authors. All suggested literature was reviewed and cross-referenced by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the source code used of the method in this work as supplementary material.

REFERENCES

Kian Ahrabian, Daniel Tarlow, Hehuimin Cheng, and Jin LC Guo. Software engineering event modeling using relative time in temporal knowledge graphs. *arXiv preprint arXiv:2007.01231*, 2020.

Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12(2), 2015.

Wei Chen, Yuting Wu, Shuhan Wu, Zhiyu Zhang, Mengqi Liao, Youfang Lin, and Huaiyu Wan. Cogntke: A cognitive temporal knowledge extrapolation framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 14815–14823, 2025.

Yuanning Cui, Zequn Sun, and Wei Hu. A prompt-based knowledge graph foundation model for universal in-context reasoning. *Advances in Neural Information Processing Systems*, 37:7095–7124, 2024.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Michael Galkin, Zhaocheng Zhu, Hongyu Ren, and Jian Tang. Inductive logical query answering in knowledge graphs. *Advances in neural information processing systems*, 35:15230–15243, 2022.

Julia Gastinger, Timo Sztyler, Lokesh Sharma, Anett Schuelke, and Heiner Stuckenschmidt. Comparing apples and oranges? on the evaluation of methods for temporal knowledge graph forecasting. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 533–549. Springer, 2023.

Julia Gastinger, Shenyang Huang, Mikhail Galkin, Erfan Loghmani, Ali Parviz, Farimah Poursafaei, Jacob Danovitch, Emanuele Rossi, Ioannis Koutis, Heiner Stuckenschmidt, et al. Tgb 2.0: A benchmark for learning on temporal knowledge graphs and heterogeneous graphs. *Advances in neural information processing systems*, 37:140199–140229, 2024a.

Julia Gastinger, Christian Meilicke, Federico Errica, Timo Sztyler, Anett Schuelke, and Heiner Stuckenschmidt. History repeats itself: a baseline for temporal knowledge graph forecasting. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pp. 4016–4024, 2024b.

Deborah J Gerner, Philip A Schrodt, Omür Yilmaz, and Rajaa Abu-Jabr. Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association, New Orleans*, 35, 2002.

Andrew Halterman, Benjamin E Bagozzi, Andreas Beger, Phil Schrodt, and Grace Scraborough. Plover and polecat: A new political event ontology and dataset. In *International Studies Association Conference Paper*, 2023.

Zhen Han, Peng Chen, Yunpu Ma, and Volker Tresp. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International conference on learning representations*, 2020.

Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. Learning neural ordinary equations for forecasting future links on temporal knowledge graphs. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 8352–8364, 2021.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36, 2023.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1269–1278, 2019.

Ningyuan Li, E Haihong, Shi Li, Mingzhi Sun, Tianyu Yao, Meina Song, Yong Wang, and Haoran Luo. Tr-rules: Rule-based model for link forecasting on temporal knowledge graph considering temporal redundancy. In *Findings of the association for computational linguistics: EMNLP 2023*, pp. 7885–7894, 2023.

Yujia Li, Shiliang Sun, and Jing Zhao. Tirgn: Time-guided recurrent graph network with local-global historical patterns for temporal knowledge graph reasoning. In *Ijcai*, pp. 2152–2158, 2022a.

Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. Temporal knowledge graph reasoning based on evolutional representation learning. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 408–417, 2021.

Zixuan Li, Saiping Guan, Xiaolong Jin, Weihua Peng, Yajuan Lyu, Yong Zhu, Long Bai, Wei Li, Jiafeng Guo, and Xueqi Cheng. Complex evolutional pattern learning for temporal knowledge graph reasoning. In *Proceedings of the 60th annual meeting of the association for computational linguistics (volume 2: short papers)*, pp. 290–296, 2022b.

Qika Lin, Jun Liu, Rui Mao, Fangzhi Xu, and Erik Cambria. Techs: Temporal logical graph networks for explainable extrapolation reasoning. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 1281–1293, 2023.

Junnan Liu, Qianren Mao, Weifeng Jiang, and Jianxin Li. Knowformer: Revisiting transformers for knowledge graph reasoning. In *International Conference on Machine Learning*, pp. 31669–31690. PMLR, 2024.

Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. Tlogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 4120–4127, 2022.

Yuxuan Liu, Yijun Mo, Zhengyu Chen, and Huiyu Liu. Loge-net: Logic evolution network for temporal knowledge graph forecasting. In *International Conference on Artificial Neural Networks*, pp. 472–485. Springer, 2023.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2395–2405, 2018.

Erfan Loghmani and MohammadAmin Fazli. Effect of choosing loss function when using t-batching for representation learning on dynamic networks. *Information Sciences*, pp. 122755, 2025.

L Luo, YF Li, G Haffari, and S Pan. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *ICLR 2024: The Twelfth International Conference on Learning Representations*. ICLR, 2024.

Xin Mei, Libin Yang, Xiaoyan Cai, and Zuowei Jiang. An adaptive logical rule embedding model for inductive reasoning over temporal knowledge graphs. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 7304–7316, 2022.

Amirhossein Nadiri and Frank W Takes. A large-scale temporal analysis of user lifespan durability on the reddit social media platform. In *Companion Proceedings of the Web Conference 2022*, pp. 677–685, 2022.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35: 32928–32941, 2022.

H Ren, W Hu, and J Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *International Conference on Learning Representations (ICLR)*, 2020.

Hongyu Ren and Jure Leskovec. Beta embeddings for multi-hop logical reasoning in knowledge graphs. *Advances in Neural Information Processing Systems*, 33:19716–19726, 2020.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *CoRR*, abs/2006.10637, 2020.

Grace I Scarborough, Benjamin E Bagozzi, Andreas Beger, John Berrie, Andrew Halterman, Philip A Schrodt, and Jevon Spivey. Polecat weekly data. *Harvard Dataverse*, 62:2023, 2023.

Priyanka Sen, Sandeep Mavadia, and Amir Saffari. Knowledge graph-augmented language models for complex question answering. In *Proceedings of the 1st Workshop on Natural Language Reasoning and Structured Explanations (NLRSE)*, pp. 1–8, 2023.

Andrew Shilliday, Jennifer Lautenschlager, et al. Data for a worldwide icews and ongoing research. *Advances in Design for Cross-Cultural Activities*, 455, 2012.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Chenyan Xiong, Russell Power, and Jamie Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pp. 1271–1279, 2017.

Ze Zhao, He Yuhui, Lyuwen Wu, Gu Tang, Bin Lu, Xiaoying Gan, Luoyi Fu, Xinbing Wang, and Chenghu Zhou. Inductive reasoning for temporal knowledge graphs with emerging entities. In *The Fourteenth International Conference on Learning Representations*, 2026.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.

A Datasets Statistics and Description

We use the following TKG and THG datasets from TGB2.0, which span diverse scales, temporal granularities, and domains. Table 5 summarizes the full statistics of the datasets used. In Table 5, Induct. Test Nodes denotes the proportion of nodes in the test set that were not observed during training. DRec, the Direct Recurrency Degree, is the fraction of test temporal quadruplets (s, r, o, t^+) for which the exact same triple (s, r, o) appeared at the immediately preceding timestamp. Rec, the Recurrency Degree, is the fraction of test temporal quadruplets (s, r, o, t^+) for which the same triple (s, r, o) appeared at any earlier timestamp. Con denotes consecutiveness, measuring how long quadruplets persist across consecutive timestamps. Mean Edges/Ts. and Mean Nodes/Ts. denote the average number of edges and active nodes per timestamp, respectively. We next provide a brief description of each dataset, highlighting its source, domain, temporal coverage, and prediction task.

- `Smallpedia` (Vrandečić & Krötzsch, 2014): A knowledge graph built from the Wikidata dataset that contains facts paired with Wikipedia pages. Each fact connects two entities through an explicit relation, represented as an edge type. Only entities from Wikidata with ID smaller than a million are included. Its temporal relations describe either point-in-time events or facts with duration. The task is to predict future facts.
- `Polecat` (Scarborough et al., 2023; Halterman et al., 2023): Based on the POLECAT dataset—POLitical Event Classification, Attributes, and Types, that provides a large-scale record of coded interactions between socio-political actors, covering both cooperative and hostile actions. Built on the PLOVER ontology, POLECAT analyzes multilingual news reports in seven languages from across the globe and converts them into structured political event records. The dataset is spanning from January 2018 to December 2022, and its task is to forecast future political events between actors.
- `ICEWS` (Gerner et al., 2002; Shilliday et al., 2012; Boschee et al., 2015) A large-scale temporal knowledge graph constructed from the ICEWS Coded Event Data that records political events between political actors. Each event is categorized using the CAMEO taxonomy, a fine-grained event ontology developed for the analysis of political conflict, mediation, and international interactions. The dataset is spanning from 1995 to 2022, and its prediction task is to forecast future interactions between political actors.
- `Wikidata` (Vrandečić & Krötzsch, 2014): A Large-scale dataset constructed from the Wikidata knowledge graph and serves as a large-scale extension of `SmallPedia`. Its temporal edges correspond to time-stamped relations between Wikidata entities, extracted from pages whose identifiers fall within the first 32 million Wikidata IDs. The prediction task of the dataset is to predict facts that will emerge in the future.
- `Software` (Ahrabian et al., 2020): This dataset is constructed from GitHub activity data collected via GH Arxiv. Only nodes with at least 10 incident edges are included, yielding a heterogeneous temporal graph with 4 node types and 14 relation types. The resulting dataset covers GitHub activity from January 2024. The predictive task is to forecast an entity's future interactions, such as which pull request a user will close next.
- `Forum` (Nadiri & Takes, 2022) is constructed from the Reddit user–subreddit interaction network, comprising two node types, users and subreddits, and two relation types: user-to-user replies and user-to-subreddit posts. The dataset contains timestamped interactions in January 2014. The associated prediction task is to forecast, for a given user at a particular time, the next user or subreddit with which they will interact.
- `GitHub` (Ahrabian et al., 2020): is constructed from GitHub activity records collected via GH Arxiv. The dataset captures a large-scale heterogeneous software-development network from March 2024, covering a different temporal period from the Software dataset. It contains four entity types—users, pull requests, issues, and repositories—and 14 relation types representing interactions among them. Nodes with degree at most two are not included in this dataset. The learning task in this dataset is to predict the next activity associated with a given entity.
- `MyKet` (Loghmani & Fazli, 2025): This dataset is derived from the MyKet Android application marketplace and captures user–app interactions over a period of six months and two

	Temporal Knowledge Graphs (tkg1-)				Temporal Heterogeneous Graphs (thg1-)			
Dataset	SmallPedia	Polecat	ICEWS	Wikidata	Software	Forum	GitHub	MyKet
Domain	knowledge	political	political	knowledge	software	social.	software	interac.
# Quadruples	550,376	1,779,610	15,513,446	9,856,203	1,489,806	23,757,707	17,499,577	53,632,788
# Nodes	47,433	150,931	87,856	1,226,440	681,927	152,816	5,856,765	1,530,835
# Edge Types	283	16	391	596	14	2	14	2
# Node Types	–	–	–	–	4	2	4	2
# Timesteps	125	1,826	10,224	2,025	689,549	2,558,457	2,510,415	14,828,090
Granularity	year	day	day	year	second	second	second	second
Induct. Test Nodes	0.26	0.12	0.05	0.34	0.13	0.02	0.14	0.01
DRec	0.71	0.07	0.11	0.61	0.00	0.00	0.00	0.00
Rec	0.72	0.43	0.63	0.61	0.10	0.63	0.01	0.37
Con	5.82	1.07	1.14	5.05	1.00	1.00	1.00	1.00
Mean Edges/Ts.	4,403.01	974.59	1,516.91	4,867.26	0.56	8.87	6.54	3.15
Mean Nodes/Ts.	5,289.16	550.60	1,008.65	5,772.16	0.86	12.96	9.77	6.24

Table 5: Statistics of temporal knowledge graphs and temporal heterogeneous graphs from TGB 2.0.

weeks. Each relation represents a user-initiated installation or update event within the marketplace. The prediction task is to forecast the next application that a given user is likely to install or update.

B Additional Implementation Details

We used standard hyperparameters consistent with prior work: batch size 200, the Adam optimizer, and a learning rate of $5 \cdot 10^{-4}$ for up to 50 epochs. Following the TGB 2.0 protocol, we selected the best validation checkpoint for evaluation. We tuned the two regularization coefficients over $0, 0.05, 0.1, 0.15, 0.2$. For RF-Logic, we tuned the number of negative entities sampled during training over $2, 5, 10, 20$, where smaller values reduce memory consumption. For each benchmark, we used the largest fixed memory size k that fit within 40GB of GPU memory and could still contribute non-negligibly under the recency decay. For example, when $\lambda = 1$, the maximum contribution of entities at the memory boundary to the feature descriptor is 2^{-19}, since the decay follows a geometric series. Each benchmark run was conducted on a single NVIDIA RTX A6000 GPU and a single AMD Ryzen 9 7900X 12-Core Processor

We next describe the GPU implementation used to extract and process the two descriptor blocks from memory.

Efficient computation of 1-hop features. The 1-hop features are computed by converting the raw entity memories of the current batch into a dense relation-indexed representation. As described in Section 3, the memory module stores, for every entity, two length-k arrays: one containing the destination objects of the most recent outgoing interactions and one containing the corresponding relation identifiers. For a batch of B query subjects, we gather the corresponding memory rows, obtaining destination and relation tensors in $\mathbb{Z}^{B\times k}$. The key implementation step constructs a tensor

$$C \in \mathbb{Z}^{B\times|\mathcal{R}|\times m},$$

where m is the maximum number of relation-local entries retained per relation. For each row, the algorithm forms a composite sorting key $b_{i,j}\cdot k+j$, where $b_{i,j}$ is the relation id at memory position j. Since memory positions are ordered from most to least recent, sorting by this key groups entries by relation while preserving temporal order within each relation group. The algorithm then identifies relation-boundary changes, uses a cumulative maximum over boundary positions to compute the relation-local rank of each entry, and scatters each destination object into $C[\text{batch}, \text{relation}, \text{rank}]$. Entries with invalid or padded relation ids, as well as entries whose relation-local rank exceeds m, are ignored and remain equal to the padding sentinel. This realizes the relation-specific histories $\mathcal{H}_{s,r'}(t^+)$ for all subjects and relations using only batched sorting, masking, cumulative reductions, and scatter assignments, avoiding explicit loops over entities or relations. Given C, all candidate objects for the same query batch are processed in parallel. A candidate tensor of shape $B \times D$ is broadcast against C, producing a match tensor of shape $B \times D \times |\mathcal{R}| \times m$. This tensor is multiplied by the precomputed decay vector $2^{-\lambda i}$, summed over the relation-local memory dimension, and divided by cumulative decay normalizers indexed by the number of valid entries in each relation-

local history. The output is the full 1-hop descriptor tensor in $\mathbb{R}^{B\times D\times|\mathcal{R}|}$, containing $d^{(1)}_{s,o}[r']$ for every query subject, candidate object, and historical relation.

Efficient computation of shared-object features. The shared-object features are computed from the same outgoing entity memories, but operate on unique relation–object pairs rather than relation-local sequences. For a batch with B query subjects and D candidate objects per subject, the implementation gathers the raw memories of the subjects into tensors of shape $B \times k$ and the raw memories of all candidates into tensors of shape $B \times D \times k$. Each valid memory entry (r', x) is encoded as a single integer key $r' \cdot S + x$, where the stride S is larger than the maximum entity id, making the encoding injective over relation–object pairs. Padded entries are mapped to a sentinel key larger than every valid key. The subject and candidate key tensors are sorted along the memory dimension. Consecutive equal keys are marked as duplicates, and only the first occurrence of each key is retained, yielding GPU-resident deduplicated representations of the sets $\mathcal{U}_{s,r'}(t^+)$ and $\mathcal{U}_{o,r'}(t^+)$ across all relations. The per-relation set sizes used in the denominators of equation 5 are computed by decoding the relation id from each retained key and applying `scatter_add` into relation bins, producing subject counts of shape $B\times|\mathcal{R}|$ and candidate counts of shape $B\times D\times|\mathcal{R}|$. To compute intersections without explicitly materializing object sets, the deduplicated candidate keys are flattened over the candidate dimension and searched inside the sorted deduplicated subject keys using batched `searchsorted`. A match is counted only when the retrieved subject key equals the candidate key and the key is not the sentinel. Matched keys are decoded back to relation ids and accumulated with another `scatter_add`, producing the intersection tensor

$$|\mathcal{U}_{s,r'}(t^+)\cap\mathcal{U}_{o,r'}(t^+)|$$

for all $B \times D$ candidate pairs and all historical relations simultaneously. Finally, the implementation computes the two normalized ratios, clamps empty denominators to avoid division by zero, and averages them to obtain $d^{(2)}_{s,o}[r']$. This design reduces shared-object rule evaluation to GPU-parallel primitives: gathering, integer key encoding, sorting, deduplication, batched binary search, scatter-based counting, and vectorized normalization. This allows the model to score many candidate pairs without constructing pairwise object sets or using nested loops over candidates, objects, and relations.

C Additional results

In Tables 6 and 7 we provide the results of RF-LOGIC compared to the baselines on TGB 2.0 TKG datasets measured in Hits@3 and Hits@10 respectively. In Tables 8 and 9 we provide the results of RF-LOGIC compared to the baselines on TGB 2.0 THG datasets measured in Hits@3 and Hits@10 respectively.

Table 6: HITS@3 on TGB knowledge graph datasets (higher is better). Boldface marks the best method.

Method	Smallpedia		Polecat		ICEWS		Wikidata	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.547	0.442	0.057	0.055	0.08	0.08	**0.676**	**0.575**
EdgeBank$_{\infty}$	0.489	0.416	0.045	0.043	0.002	0.001	**0.676**	**0.575**
RecB$_{\text{train}}$	0.690	0.669	0.225	0.221	-	-	OOT	OOT
RecB$_{\text{default}}$	0.586	0.535	0.186	0.184	0.294	0.226	OOT	OOT
CognTKE	0.547	0.5264	0.325	0.317	OOM	OOM	OOM	OOM
TransFIR	0.340±0.068	0.224±0.050	0.251±0.006	0.217±0.004	OOM	OOM	OOM	OOM
RE-GCN	0.656±0.003	0.625±0.001	0.219±0.002	0.196±0.003	0.256±0.002	0.201±0.002	OOM	OOM
CEN	0.683±0.001	0.659±0.001	0.237±0.001	0.215±0.001	0.271±0.001	0.213±0.001	OOM	OOM
TLogic	0.696±0.001	0.680±0.001	0.275±0.001	0.267±0.001	**0.327**±0.001	0.216±0.001	OOT	OOT
RF-LOGIC (ours)	**0.695**±0.001	**0.683**±0.001	**0.503**±0.001	**0.491**±0.001	0.316±0.001	**0.298**±0.001	0.656 ±0.001	0.560±0.001

D Further Interpretability Analysis

In Section 4 we have seen a visualization of the learned weights of RF-LOGIC, and saw that it can teach us significantly about how RF-LOGIC makes its decisions to rank different objects. Next, we dive deeply into three representative examples that demonstrate that not only we can visually

Table 7: HITS@10 on TGB knowledge graph datasets (higher is better). Boldface marks the best method.

Method	Smallpedia		Polecat		ICEWS		Wikidata	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.672	0.566	0.126	0.118	0.042	0.043	**0.692**	**0.595**
EdgeBank$_{\infty}$	0.611	0.562	0.099	0.093	0.016	0.010	**0.692**	**0.595**
RecB$_{\text{train}}$	**0.727**	**0.716**	0.316	0.316	-	-	OOT	OOT
RecB$_{\text{default}}$	0.685	0.651	0.261	0.263	0.398	0.322	OOT	OOT
CognTKE	0.591	0.5573	0.444	0.442	OOM	OOM	OOM	OOM
TransFIR	0.467±0.045	0.335±0.042	0.438±0.011	0.395±0.008	OOM	OOM	OOM	OOM
RE-GCN	0.703±0.001	0.675±0.001	0.322±0.003	0.298±0.002	0.399±0.002	0.330±0.002	OOM	OOM
CEN	0.727±0.001	0.706±0.001	0.343±0.001	0.320±0.001	0.409±0.001	0.331±0.001	OOM	OOM
TLogic	0.717±0.001	0.707±0.001	0.383±0.001	0.378±0.001	**0.483**±0.001	0.306±0.001	OOT	OOT
RF-LOGIC (ours)	0.708±0.001	0.707±0.001	**0.628**±0.001	**0.624**±0.001	0.480±0.001	**0.458**±0.001	0.658±0.001	0.562±0.001

Table 8: HITS@3 on TGB temporal heterogeneous graph datasets (higher is better). Boldface marks the best method.

Method	Software		forum		Github		Myket	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.279	0.289	0.535	0.529	0.292	0.312	0.173	0.169
EdgeBank$_{\infty}$	0.397	0.4487	0.622	0.627	0.356	0.366	0.373	0.403
RecB$_{\text{default}}$	0.102	0.110	-	-	OOT	OOT	OOT	OOT
TGN†	0.052±0.004	0.061±0.007	0.680±0.040	0.609±0.058	OOM	OOM	OOM	OOM
TGN$^{\dagger}_{\text{edge-type}}$	0.213±0.006	0.252±0.006	0.680±0.040	0.609±0.058	OOM	OOM	OOM	OOM
RF-LOGIC (ours)	**0.435**±0.001	**0.498**±0.001	**0.751**±0.001	**0.753**±0.001	**0.394**±0.001	**0.426**±0.001	**0.373**±0.001	**0.403**±0.001

understand the model's decisions but also to see that the decisions made by the model are deeply data driven.

Relation clusters in Polecat. Unlike datasets such as SmallPedia and Wikidata, the largest learned coefficients for Polecat are concentrated off the diagonal and form two clear relation clusters, R0–R15 and R16–R31. TGB 2.0 augments each TKG dataset with inverse relations r^{-1}: for every event (s, r, o, t), it also adds (o, r^{-1}, s, t). This augmentation explains the duplicated behavior observed in the learned weights for SmallPedia and Wikidata. In Polecat, however, RF-LOGIC learns a more asymmetric structure: a relation r can be informative for predicting its inverse relation r^{-1}, but not necessarily vice versa. To connect this learned structure to the temporal graph dynamics, Figure 4 reports, for each query relation r_q and memory relation r_m, the hit rate: the fraction of queries (s, r_q, o, t) for which the most recent tuple of relation r_m matches the queried entity pair (s, o). The left plot shows this hit rate directly, while the right plot shows its log-lift over a random baseline, $\log_2(\text{hit rate}/\text{baseline})$. The resulting block structure closely matches the learned weights of RF-LOGIC, indicating that the model captures genuine relation-specific temporal dependencies in Polecat rather than merely a general recency bias.

Negative diagonal weights in GitHub. In Section 4, we observed that on the GitHub dataset, RF-LOGIC learns a sparse weight matrix whose diagonal entries are mostly negative, consistent with the negation rule introduced in our model. We now show that these learned weights reflect the dynamic behavior of the graph. In Figure 5, the ratio of distinct objects to events for each $(\text{subject}, \text{relation})$ pair is concentrated near one in GitHub, with an event-weighted mean of 0.990. This indicates that most events introduce a new object for the same subject–relation pair, i.e., the graph exhibits a high degree of "surprise." In contrast, SmallPedia has an event-weighted mean of only 0.126, reflecting substantially more object reuse. Similarly, Figure 6 shows that exact $(\text{subject}, \text{relation}, \text{object})$ repetitions are rare in GitHub, with an event-weighted repetition rate of 0.010, whereas SmallPedia exhibits a much higher repetition rate of 0.874. These statistics explain why RF-LOGIC assigns negative diagonal weights on GitHub: previously observed triples are unlikely to repeat and therefore provide negative evidence for future events. Conversely, the repetitive dynamics of SmallPedia make positive diagonal weights beneficial.

Shared-object reliance in Forum. In Section 4, we observed that the 1-hop features are not used to predict target relation R0 in Forum. Instead, RF-LOGIC relies on shared-object features. We now show that this learned behavior is aligned with the temporal dynamics of the dataset. In Forum, re-

Table 9: HITS@10 on TGB temporal heterogeneous graph datasets (higher is better). Boldface marks the best method.

Method	Software		forum		Github		Myket	
	Val.	Test	Val.	Test	Val.	Test	Val.	Test
EdgeBank$_{tw}$	0.283	0.291	0.535	0.529	0.292	0.312	0.173	0.169
EdgeBank$_{\infty}$	0.407	0.455	0.624	0.629	0.357	0.367	0.373	0.403
RecB$_{\text{default}}$	0.160	0.167	-	-	OOT	OOT	OOT	OOT
TGN†	0.141±0.007	0.164±0.013	0.881±0.004	0.864±0.008	OOM	OOM	OOM	OOM
TGN$_{\text{edge-type}}$ †	0.355±0.007	0.408±0.010	**0.881**±0.004	**0.864**±0.008	OOM	OOM	OOM	OOM
RF-LOGIC (ours)	**0.435**±0.001	**0.498**±0.001	0.806±0.001	0.809±0.001	**0.394**±0.001	**0.426**±0.001	**0.373**±0.001	**0.403**±0.001

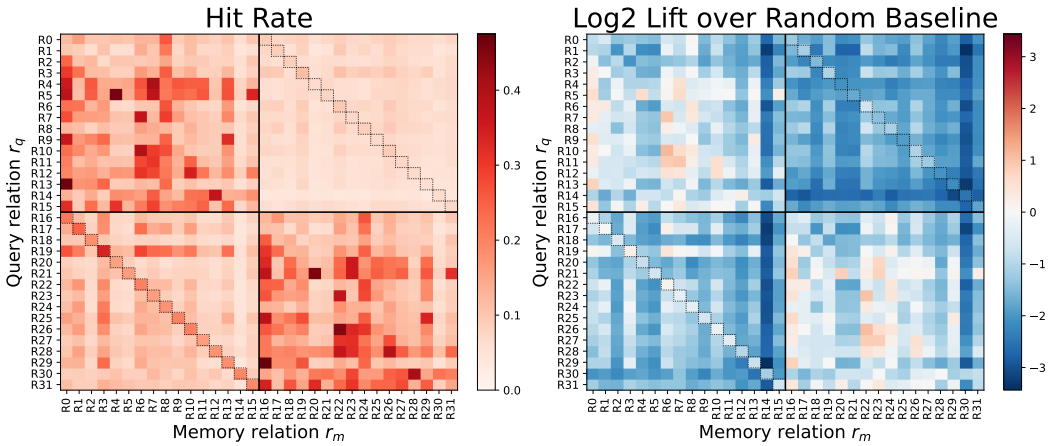

Figure 4: Polecat hit rate and hit rate lifted with random baseline

lation R0 represents user–user interactions, whereas relation R1 represents user–thread interactions. This typing helps explain the zero weight assigned by RF-LOGIC to the anti-diagonal entries of the 1-hop relation-weight matrix: these cross-relation 1-hop patterns do not directly provide reliable evidence for R0 prediction. To examine the shared-object signal empirically, Figure 7 reports the probability that a future R0 edge $(s, \mathrm{R0}, o)$ is positive as a function of the number of objects shared by s and o under previous R0 and R1 relations. The probabilities are estimated using one positive destination and 20 sampled negative destinations per query. We also report the corresponding $\log_2$ lift over the uniform baseline probability $1/21$. The figure shows that future R0 interactions are strongly correlated with shared objects under both R0 and R1. This is consistent with the semantics of Forum: users who interacted with the same users are more likely to interact with each other, and users who posted in the same threads are also more likely to interact. Thus, the shared-object rule captures a meaningful higher-order signal that is missed by the 1-hop relation weights alone.

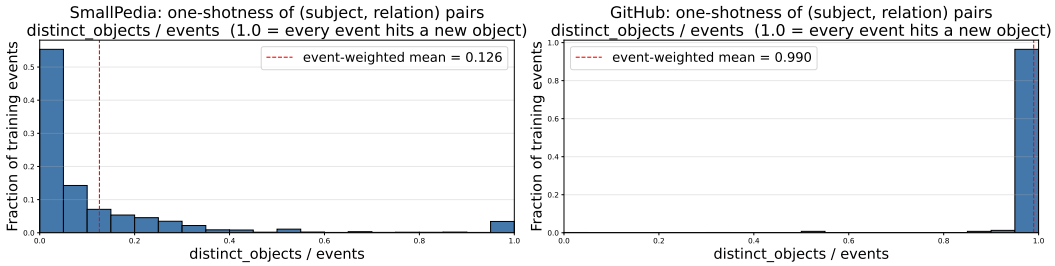

Figure 5: RF-LOGIC SmallPedia compared to GitHub in the ratio of number of distinct objects per (subject,relation) event.

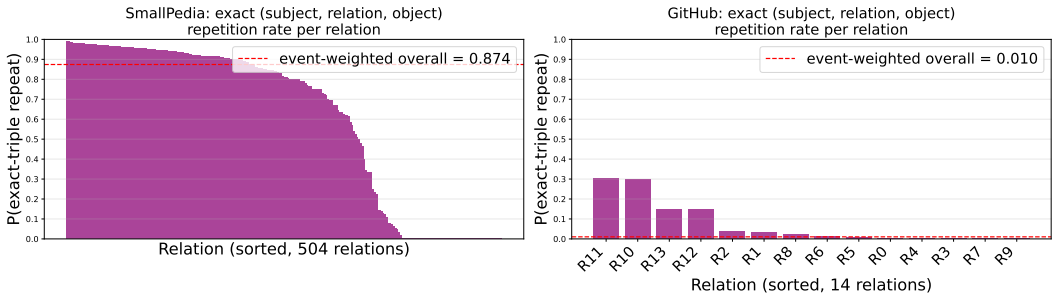

Figure 6: SmallPedia compared to GitHub in the repetition rate.

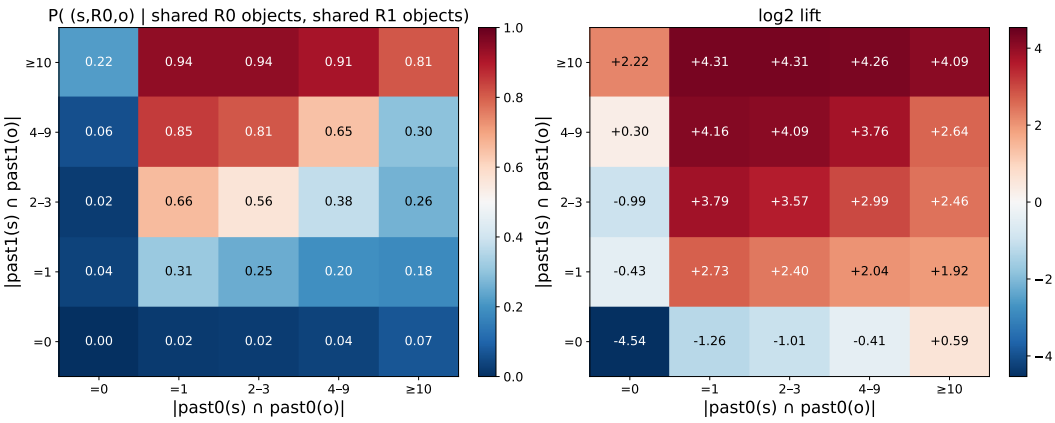

Figure 7: Conditional R0 occurrence probability and lifted probability

E ABLATION STUDY

We ablate two modeling choices in RF-LOGIC: the recency-decay parameter λ used in equation 3, and the training objective used to optimize the scoring model. We use `Polecat` and `Software` as representative datasets because they differ both in domain and temporal structure: `Polecat` is a daily political-event TKG, while `Software` is a second-resolution heterogeneous interaction graph. Unless otherwise stated, all other hyperparameters are fixed to the configuration used in the main experiments. For each setting, we select the checkpoint by validation MRR and report the corresponding test metrics.

E.1 RECENCY-DECAY HYPERPARAMETER

The parameter λ controls the trade-off between frequency and recency. When $\lambda = 0$, equation 3 reduces to a relation-local frequency feature over the stored history. Larger values place progressively more mass on the most recent interactions. The results in Tables 10 and 11 show that this trade-off is dataset dependent.

On `Polecat`, performance improves steadily as λ increases: test MRR rises from 0.278 at $\lambda = 0$ to 0.424 at $\lambda = 2.0$, and Hits@1 rises from 0.186 to 0.319. This indicates that recent relation-local evidence is substantially more predictive than unweighted historical frequency for political-event forecasting. The validation and test curves also follow the same trend, suggesting that the gain is not an artifact of validation selection.

On `Software`, the best setting is a mild recency bias ($\lambda = 0.25$), and the full sweep is much flatter: test MRR varies only from 0.494 to 0.498. This behavior is consistent with the low exact-recurrence statistics of `Software` in Table 5: strong recency decay can discard useful frequency information, while a small decay still helps distinguish the most relevant recent interactions.

Table 10: Recency-decay ablation on `Polecat`. Bold indicates the best value before rounding within each metric column.

λ	Validation				Test			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
0.00	0.293	0.485	0.327	0.200	0.278	0.468	0.311	0.186
0.25	0.359	0.595	0.416	0.247	0.345	0.586	0.403	0.232
0.50	0.390	0.617	0.456	0.277	0.379	0.609	0.443	0.265
0.75	0.407	0.623	0.475	0.297	0.396	0.617	0.463	0.285
1.00	0.417	0.626	0.487	0.309	0.407	0.621	0.474	0.298
1.25	0.424	0.627	0.494	0.318	0.414	0.623	0.482	0.306
1.50	0.429	0.628	0.498	0.323	0.419	0.624	0.486	0.312
1.75	0.432	0.628	0.501	0.328	0.422	0.624	0.489	0.316
2.00	**0.434**	**0.628**	**0.503**	**0.331**	**0.424**	**0.624**	**0.491**	**0.319**

E.2 TRAINING OBJECTIVE

We also compare the pairwise sigmoid ranking loss in equation 6 with two standard alternatives: a sampled cross-entropy objective over the positive object and sampled negatives, and a margin-based ranking objective. Table 12 reports the comparison on the same two datasets. The pairwise sigmoid objective gives the best test MRR on both datasets. On `Polecat`, cross-entropy obtains slightly higher Hits@1, but this comes with lower MRR, Hits@3, and Hits@10, suggesting that the pairwise sigmoid surrogate provides a better trade-off for the ranking metrics used in TGB2.0. On `Software`, all objectives are close, but the pairwise sigmoid loss remains consistently strongest.

F IMPACT STATEMENT

This work aims to improve the practicality, scalability, and interpretability of temporal knowledge graph forecasting. By representing predictions through recency-frequency temporal logic rules,

Table 11: Recency-decay ablation on `Software`. Bold indicates the best value before rounding within each metric column.

λ	Validation				Test			
	MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
0.00	0.433	0.435	0.434	0.430	0.496	0.498	0.498	0.492
0.25	**0.435**	**0.435**	**0.435**	**0.433**	**0.498**	**0.498**	**0.498**	**0.496**
0.50	0.434	0.434	0.434	0.432	0.497	0.497	0.497	0.495
0.75	0.433	0.433	0.433	0.432	0.497	0.496	0.496	0.495
1.00	0.432	0.432	0.432	0.431	0.496	0.496	0.496	0.494
1.25	0.432	0.431	0.431	0.430	0.495	0.495	0.495	0.494
1.50	0.432	0.431	0.431	0.430	0.495	0.494	0.494	0.494
1.75	0.431	0.430	0.430	0.430	0.495	0.494	0.494	0.493
2.00	0.431	0.430	0.430	0.430	0.494	0.493	0.493	0.493

Table 12: Training-objective ablation. Each row reports validation metrics and the corresponding test metrics from the checkpoint selected by validation MRR. Bold indicates the best value before rounding within each dataset block and metric column.

Dataset	Objective	λ	Validation				Test			
			MRR	Hits@10	Hits@3	Hits@1	MRR	Hits@10	Hits@3	Hits@1
`Polecat`	Pairwise sigmoid	2.0	**0.434**	**0.628**	**0.503**	0.331	**0.424**	**0.624**	**0.491**	0.319
`Polecat`	Cross-entropy	2.0	0.433	0.613	0.497	**0.337**	0.421	0.608	0.484	**0.323**
`Polecat`	Margin ranking	2.0	0.422	0.618	0.488	0.320	0.412	0.613	0.476	0.309
`Software`	Pairwise sigmoid	0.0	**0.433**	**0.435**	**0.434**	**0.430**	**0.496**	**0.498**	**0.498**	**0.492**
`Software`	Cross-entropy	0.0	0.432	0.433	0.433	0.428	0.494	0.496	0.496	0.491
`Software`	Margin ranking	0.0	0.432	0.433	0.433	0.429	0.494	0.495	0.495	0.491

RF-LOGIC can provide transparent evidence for its forecasts while scaling to large temporal and heterogeneous graphs under realistic memory and runtime constraints. These properties may benefit applications that rely on dynamic relational prediction, including recommendation systems, software-development analytics, knowledge-base completion, geopolitical event forecasting, and healthcare-related temporal reasoning. At the same time, forecasting future interactions or relations can raise societal concerns when applied to sensitive domains. Potential risks include amplifying biases present in historical graph data, privacy leakage from temporal interaction histories, misuse for surveillance or profiling, and overreliance on incorrect predictions in high-stakes decision-making. RF-LOGIC does not introduce new data sources or autonomous decision-making capabilities; its ethical implications depend primarily on the datasets and downstream deployment contexts. Responsible use should therefore include careful dataset governance, privacy protections, bias and fairness evaluation, uncertainty-aware interpretation of forecasts, and human oversight in sensitive applications.